MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1, 2016/2017

TCS 3311 – COMPILER DESIGN/ TCP2451 – PROGRAMMING LANGUAGE TRANSLATION

(All Sections / Groups)

17 October 2016
9 AM – 11 AM.
(2 Hours)

INSTRUCTION TO STUDENT

1. Answer **ALL** questions.
2. This question paper has 5 printed pages excluding the front cover.
3. Please print all your answers in the answer booklet provided.

QUESTION 1

A programming language designer is creating a new language with the following tokens:

Command – three letter codes, where the middle letter is a vowel
Letter Initialized Number – a single letter followed by an integer number
Identifier – a letter followed optionally by one or more letters or digits

All letters used in the above are in small letter form.

(a) Write down a regular expression for each of the three token patterns.

[3 marks]

(b) Draw an NFA to recognize tokens based on the combination of these regular expressions. Use Thompson's construction approach.

[5 marks]

(c) Convert the NFA in Question 1(b) to a DFA using subset construction method. Show all your working and the transition table.

[7 marks]

Continued .....

QUESTION 2

Consider the following context free grammar:

P → { L }
L → S L
L → ε
S → **id** = E;
E → NF
F → +NF
F → ε
N → (E) | **num**

P is the start symbol, and the terminals are ; { } **id** = + ( ) **num**.

(a) Write the FIRST and FOLLOW sets for all the non-terminals in this grammar.

[5 marks]

(b) Draw the LL(1) parsing table for this grammar.

[5 marks]

(c) Based on the parsing table in Question 2(b), write a recursive procedure `parse_F` that parses this grammar by recursive descent. Your procedure may be written in Java, C++, Visual Basic or pseudocode. You may assume the existence of a global variable `nexttoken` that holds the next input token, a function `match()` that reads the next token into `nexttoken`, and a function parse_error that may be called if the input is not in the language generated by the grammar.

[5 marks]

Continued .....

QUESTION 3

(a) For the following BNF grammar write a Java CUP specification file

```
<assignment> ::= <identifier> = <expression> ;
<expression> ::= <identifier> + <identifier>
                 | <identifier> - <identifier>
<identifier> :: a | b | c
```

If you had made any assumptions, write it down clearly.

[6 marks]

(b) Identify synthesized attributes and inherited attributes required for each of the non-terminals in the question 3(a) grammar to implement the following semantic descriptions.

Semantic descriptions:

- The <identifier> can be one of two types: int or float
- The type of the expression <expression> when the operand types are not the same is always float.
- The type of the expression <expression> when the operand types are the same is always of its operand.
- The type of the left side of the assignment must match the type of the right side.

Complete the following table with your answers (copy this table into your answer booklet)

Non-terminals	Synthesized attributes	Inherited attributes
<assignment>		
<identifier>		
<expression>		

[3 marks]

Continued .....

(c) Write semantic rules and conditions for the grammar in question 3(a) using the semantic descriptions given in question 3(b). Make use the synthesized attributes and inherited attributes you have identified in question 3(b).

Complete the following table with your answers (copy this table into your answer booklet). If you made any assumptions, state it clearly.

Syntax rule	Semantic rule & conditions
<assignment> ::= <identifier> = <expression>	
<expression> ::= <identifier>[2] + <identifier>[3]	
<expression> ::= <identifier>[4] - <identifier>[5]	
<identifier> ::= a | b | c	

[6 marks]

Continued .....

QUESTION 4

(a) Given the code statements:

```
        istore 1
Label_1:
        iload 1
        if_icmplt Label_2
        iconst 0
        ifeq Label_3
        iload 2
        goto Label_1
Label_2:
        iconst 1
Label_3:
        ifeq Label_2
        iload 3
        ireturn
```

Identify the basic blocks and draw the flow graph.

[5 marks]

(b) Given the following expression:

$$a * (b + c - d) + (b + c)$$

Explain how a Directed Acyclic Graph can be used to help to optimize the intermediate codes generated for this expression. Illustrate this by writing the set of optimized intermediate codes.

[5 marks]

(c) Given the following set of codes that could be generated for the statement "a := b + c", where R0 is a register and a, b, c are addressed memory locations of the variables.

Set 1	Set 2

MOV b, R0
ADD c, R0
MOV R0, a
	MOV b, a
	ADD c, a

Calculate the instruction costs for each set of instructions and discuss the differences between the two sets of instructions.

[5 marks]

End of Page